

Exploring the rationality of some syntactic merging operators (extended version)

José Luis Chacón and Ramón Pino Pérez

*Departamento de Matemáticas
Facultad de Ciencias
Universidad de Los Andes
Mérida, Venezuela
{jlchacon,pino}@ula.ve*

Abstract. Most merging operators are defined by semantics methods which have very high computational complexity. In order to have operators with a lower computational complexity, some merging operators defined in a syntactical way have been proposed. In this work we define some syntactical merging operators and explore its rationality properties. To do that we constrain the belief bases to be sets of formulas very close to logic programs and the underlying logic is defined through forward chaining rule (Modus Ponens). We propose two types of operators: arbitration operators when the inputs are only two bases and fusion with integrity constraints operators. We introduce a set of postulates inspired of postulates LS, proposed by Liberatore and Shaerf and then we analyzed the first class of operators through these postulates. We also introduce a set of postulates inspired of postulates KP, proposed by Konieczny and Pino Pérez and then we analyzed the second class of operators through these postulates.

1 Introduction

Belief merging [30,31,2,3,27,26,25,24,18] aims at combining several pieces of information when there is no strict precedence between them, unlike belief revision [1,11,17,12] where one combines two pieces of information one of which has higher priority. The agent faces several conflicting pieces of information coming from several sources of equal reliability¹, and he has to build a coherent description of the world from them. One important aspect of belief merging, differentiating this theory from belief revision -even from non-prioritized belief revision (see [13])- is the fact that n sources of information (with $n \geq 2$) are considered.

This work is about belief merging in the framework of logic-based representation of beliefs. In this framework beliefs are sets of propositional formulas.

¹ More generally the sources can have different reliability, but we will focus on the case where all the sources have the same reliability. There is already a lot to say in this case.

Many merging operators have been defined in that setting (a complete survey of logic-based merging is [19]). Most merging operators are based in semantical representations and the computational complexity of the entailment problem is at least in the second level of the polynomial hierarchy. Precisely, the problem of deciding if a formula is entailed by the revised base is in the class Π_2^P [10,28] and the fact that belief revision operators are a particular case of belief merging operators [18] say us that the last operators are complex at least concerning the entailment problem.

In recent years there has been a growing interest in studying belief revision and merging in some particular fragments of propositional logic. In particular Horn clauses [6,22,9,7], logic programs [14,8] and other more general definable fragments [5].

Some works have been done to define change operators in a syntactic way [15,4]. One interesting feature of operators defined in [4] is that the computational complexity is polynomial. In that work some revision operators and update operators are syntactically defined with a restriction of the language and the logic. Therein the only inference rule is Modus Ponens and the formulas are very close to clauses in Logic Programming. But the semantics is classical and very simple and natural. Our representation of beliefs will be close to Logic Programs but with the natural and classical semantics. In the present work we study some operators of merging for which the beliefs have this simple representation.

The first idea we explore is the use of a revision operator $*$ to define binary merging operator Δ . The key point here is trying to emulate the following equation²

$$\varphi \Delta \psi = (\varphi * \psi) \vee (\psi * \varphi) \quad (1)$$

Notice that this kind of operators have been called arbitration operators by Liberatore and Schaerf [23,24]. Actually they characterized these operators by a set of postulates of rationality. Here we adapt the postulates to the simple logic we use, we define syntactic arbitration operators and we study them to the light of Liberatore and Shaerf postulates modified.

We explore afterwards the idea of defining merging operators with integrity constraints using the following equation as a guide:

$$\Delta_\mu(\varphi_1, \dots, \varphi_n) = \begin{cases} \varphi_1 \wedge \dots \wedge \varphi_n \wedge \mu, & \text{if consistent;} \\ \bigvee (\varphi_i * \mu), & \text{otherwise.} \end{cases} \quad (2)$$

We shall define syntactic merging operators with integrity constraints adapting the previous equation to the logic of forward chaining we use; we shall define the postulates characterizing merging operators (the natural adaptation of KP postulates [18]) in this simple setting and then we shall analyze the satisfaction of these postulates by our syntactic integrity constraint merging operators.

For the two kinds of operators we will see, on one hand, that the computational complexity is polynomial. This is a considerable gain with respect to the

² The symbols φ , ψ and μ (with subscripts if necessary) denote propositional formulas which are usually used to represent beliefs.

more classical merging operators; on the other hand, some postulates are not satisfied. That is the compromise in order to have tractable operators: you gain in computational complexity but you lose some properties.

The rest of the paper is organized as follows: Section 2 contains the basic definitions and the syntactic revision operators (first defined in [4]) used later. Section 3 is devoted to study a syntactic arbitrage operator following the lines of Equation 1. Section 4 is devoted to analyze the syntactic merging operators defined following the lines of Equation 2. We finish with a section containing some concluding remarks.

2 Preliminaries

Let \mathcal{V} be a finite set of propositional variables. The elements of \mathcal{V} are called atoms. A literal (or fact) is an atom or the negation of an atom. The set of literals will be denoted *Lit*. A *rule* is a formula of the shape $l_1 \wedge l_2 \cdots \wedge l_n \rightarrow l_{n+1}$ where l_i is a literal for $i = 1, \dots, n+1$. Such a rule will be denoted $l_1, l_2, \dots, l_n \rightarrow l_{n+1}$; the part l_1, l_2, \dots, l_n is called the body of the rule and l_{n+1} is called the head of the rule. A fact l can be seen as a rule $\rightarrow l$ with empty body.

Let R and L be a finite set of rules with non empty body and a finite set of facts respectively. A program P is a set of the form $R \cup L$. In such a case we will say that the elements of R are the rules of P and the elements of L are the facts of P . The set of programs will be denoted *Prog*. Let $P = R \cup L$ be a program. We define the set of consequences by forward chaining of P , denoted $C_{fc}(P)$, as the smallest set of literals (with respect to inclusion) L' such that:

- (i) $L \subseteq L'$
- (ii) If $l_1, l_2, \dots, l_n \rightarrow l$ is in R and $l_i \in L'$ for $i = 1, \dots, n$ then $l \in L'$.
- (iii) If L' contains two opposed literals (an atom and its negation) then $L' = Lit$.

A program P is consistent if $C_{fc}(P)$ does not contain two opposed literals (or alternatively $C_{fc}(P) \neq Lit$), otherwise we say that P is inconsistent and it will be denoted $P \vdash \perp$. Let R , L and L' be a finite set of rules and two finite sets of facts respectively. L is said to be R -consistent if $R \cup L$ is consistent. L is said to be $R \cup L'$ consistent if $L \cup (R \cup L')$ is consistent. Thus, if P is a program, L is P -consistent if $L \cup P$ is consistent. Let L and P be a set of literals and a program respectively. L is said to be fc-consequence of P if $L \subseteq C_{fc}(P)$.

It is important to note that in this setting the problem of the consistency is polynomial and the problem of a literal entailment is also polynomial.

We can define a very natural hierarchy over $C_{fc}(P)$ for a program P . Let L be $C_{fc}(P)$. We define a partition of L , $L = L_0 \cup L_1 \cdots \cup L_r$, inductively in the following way: L_0 is the set of facts of program P ; L_i are the literals $l \in L \setminus \{L_0 \cup L_1 \cdots \cup L_{i-1}\}$ such that there is a rule $l_1, l_2, \dots, l_k \rightarrow l \in P$ satisfying $\{l_1, \dots, l_k\} \subset L_0 \cup L_1 \cdots \cup L_{i-1}$. Thus, in L_1 we find the literals not in L_0 and obtained by the rules of P using the literals of L_0 . In L_i we find the literals not in L_j with $0 \leq j < i$ obtained by the rules of P using the literals of $L_0 \cup \dots \cup L_{i-1}$. Since $P = R \cup L_0$, where R is a finite set of rules and L_0 is a finite set of facts, it follows that $|L| \leq |L_0| + |R|$, so each L_i is a finite set.

Example 1 Consider $P = \{a \rightarrow b; a \rightarrow c; b \rightarrow t; c \rightarrow s; t \rightarrow s; s \rightarrow w; u \rightarrow h; a; u\}$ then $L_0 = \{a, u\}$, $L_1 = \{b, c, h\}$, $L_2 = \{t, s\}$ and $L_3 = \{w\}$.

The following result whose proof is easy will be useful later:

Lemma 1 Let Q, P be programs, then $C_{fc}(Q \cup P) = C_{fc}(C_{fc}(Q \cup P) \cup P)$.

Now we shall recall some definitions of operators in [4].

Definition 1 (Exceptional sets of literals and rules) Let P be a program. A set of literals L is said to be exceptional for P if L is not P -consistent. A rule $L \rightarrow l$ in P is exceptional for P if L is exceptional for P .

Observation 1 The point that makes all computations simple is the fact that, given the definition of entailment and consistency we have, to know if a set of literals is P -consistent is polynomial.

Notice that if P is not consistent all its rules are exceptional. The following hierarchy of a program appears in [29]. It has been very useful (see for instance [21]).

Definition 2 (Base) Let P be a program. We define $(P_i)_{i \in \mathbb{N}}$ a decreasing sequence of programs in the following way: P_0 is P and P_{i+1} is the set of exceptional rules of P_i . Since P is finite, there is a first integer n_0 such that for all $m > n_0$, $P_m = P_{n_0}$. If $P_{n_0} \neq \emptyset$ we say that $\langle P_0, \dots, P_{n_0}, \emptyset \rangle$ is the base of P . If $P_{n_0} = \emptyset$, then $\langle P_0, \dots, P_{n_0} \rangle$ will be the base of P .

For instance, if we take $P = \{a \rightarrow b; b \rightarrow \neg c; \neg c \rightarrow \neg a; \neg c \rightarrow b; \neg a \rightarrow \neg b; \neg a \rightarrow \neg c\}$, it is easy to see that all the rules of P are exceptional, thus the base of P is $\langle P, \emptyset \rangle$. Now we give another (classic taxonomic) example:

Example 2 Take $P = \{m \rightarrow s; c \rightarrow m; c \rightarrow \neg s; n \rightarrow c; n \rightarrow s\}$ where m, s, c, n represent mollusk, shell, cephalopod and nautilus respectively. The base is $\langle P_0, P_1, P_2, P_3 \rangle$ where $P_0 = \{m \rightarrow s; c \rightarrow m; c \rightarrow \neg s; n \rightarrow c; n \rightarrow s\}$, $P_1 = \{c \rightarrow m; c \rightarrow \neg s; n \rightarrow c; n \rightarrow s\}$, $P_2 = \{n \rightarrow c; n \rightarrow s\}$ and $P_3 = \emptyset$.

Definition 3 (Rank function) Let P be a program and let $\langle P_0, \dots, P_n \rangle$ be its base. We define $\rho(P, \cdot) : \text{Prog} \rightarrow \mathbb{N}$, the rank function, as follows: $\rho(P, Q) = \min\{i \in \mathbb{N} : Q \text{ is } P_i\text{-consistent}\}$ if P and Q are consistent, otherwise $\rho(P, Q) = n$.

Notice that if $Q_1 \subseteq Q_2$ then $\rho(P, Q_1) \leq \rho(P, Q_2)$.

Definition 4 (Rank revision) Let P and P' be two programs. We define the rank revision operator of P by P' , denoted $P \circ_{rk} P'$, as follows:

$$P \circ_{rk} P' = P_{\rho(P, P')} \cup P'$$

That is, we take the new piece of information P' together with the first program in the base (the least exceptional) which is consistent with P' .

In order to generalize this operator we need to define the hull of a program P with respect to another program P' . Let $I_P(P')$ the set of subsets of P which are consistent with P' , contain $P_{\rho(P,P')}$ and maximal with these properties. We define $h_P : Prog \rightarrow \mathcal{P}(P)$ by letting $h_P(P') = \bigcap I_P(P')$. The computation of $I_P(P')$ is exponential.

Definition 5 (Hull revision) *Let P and P' be two programs. We define the hull revision operator of P by P' , denoted $P \circ_h P'$, as follows:*

$$P \circ_h P' = h_P(P') \cup P'$$

Observation 2 *It is easy to see, by the definitions, that $C_{fc}(P \circ_{rk} P') \subseteq C_{fc}(P \circ_h P')$. In such a case we say that \circ_h is a conservative extension of \circ_{rk} . Actually, there are examples in which the inclusion is strict.*

In order to define the extended hull revision we define first what is a flock of programs. This is simply a vector $\langle Q_0, \dots, Q_n \rangle$ where each Q_i is a program. We use the letters \mathcal{A}, \mathcal{F} (with subscripts if necessary) to denote flocks. We define the concatenation of flocks in the natural way: $\langle P_1, \dots, P_n \rangle \cdot \langle Q_1, \dots, Q_m \rangle \stackrel{def}{=} \langle P_1, \dots, P_n, Q_1, \dots, Q_m \rangle$. Suppose that \mathcal{A} is a flock, say $\mathcal{A} = \langle Q_1, \dots, Q_n \rangle$, we define $C_{fc}(\mathcal{A}) = \bigcap_{i=1}^n C_{fc}(Q_i)$. We identify a program P with the flock $\langle P \rangle$. With this identification flocks are more general objects than programs. We are going to define revision operators of flocks by programs in which the output will be a flock. First we consider $I_P(Q)$ as a flock and we give the following definition:

Definition 6 *Let P, P' be two programs. Let $I_P(P')$ be as before. We put*

$$P \circ_{eh} P' = \begin{cases} \langle H_1 \cup P', \dots, H_n \cup P' \rangle & \text{if } I_P(P') = \{H_1, \dots, H_n\} \\ P' & \text{if } I_P(P') = \emptyset \end{cases}$$

More generally, if $\mathcal{A} = \langle Q_1, \dots, Q_n \rangle$, we define

$$\mathcal{A} \circ_{eh} P = (Q_1 \circ_{eh} P) \cdot (Q_2 \circ_{eh} P) \cdots (Q_n \circ_{eh} P)$$

where the symbol \cdot denotes the concatenation of flocks.

Observation 3 *With the previous definition \circ_{eh} is a conservative extension of \circ_h , that is, $C_{fc}(P \circ_h P') \subseteq C_{fc}(P \circ_{eh} P')$. To see that, it is enough to notice that $h_P(P') \subseteq H_i$ for all $H_i \in I_P(P')$. For this reason the operator \circ_{eh} is called extended hull revision.*

For a study of AGM postulates satisfied by the previous operators we suggest the reader see the work in [4].

3 Syntactic arbitrage operators

Liberatore and Schaerf [23,24] define a kind of merging operators called arbitrage operators (or commutative revision operators). They consider beliefs that are represented by propositional formulas built in a finite propositional language. Their operators, denoted \diamond , map two belief bases in a new belief base and they are characterized by a set of postulates of rationality. Actually, Liberatore and Schaerf [24] prove that if \circ is a revision operator (see [16]) then \diamond defined by letting $\varphi \diamond \mu = (\varphi \circ \mu) \vee (\mu \circ \varphi)$ is an arbitrage operator.

Inspired by the previous ideas and having syntactic revision operators we are going to define syntactic arbitrage operators. We shall redefine the postulates of arbitrage in the syntactical framework and we shall analyse the relationships between our operators and these new postulates.

In order to define the syntactic operators and to state the new postulates, we must provide operators simulating over programs the conjunction and the disjunction over formulas respectively. Thus, the operator $\odot : Prog^2 \rightarrow \mathcal{P}(Lit)$ (simulating the conjunction) is defined by:

$$P_1 \odot P_2 = C_{fc}(P_1 \cup P_2)$$

And the operator $\oplus : Prog^2 \rightarrow \mathcal{P}(Lit)$ (simulating the disjunction) is defined by:

$$P_1 \oplus P_2 = C_{fc}(P_1) \cap C_{fc}(P_2)$$

The operator \odot over programs corresponds to operator \wedge over formulas and the operator \oplus over programs corresponds to operator \vee .

Now we define three syntactic arbitrage operators of the shape $\diamond : Prog^2 \rightarrow \mathcal{P}(Lit)$ in the following way:

$$P_1 \diamond P_2 = (P_1 \star P_2) \oplus (P_2 \star P_1) \quad (3)$$

where $\star \in \{\circ_{rk}, \circ_h, \circ_{eh}\}$ with $\circ_{rk}, \circ_h, \circ_{eh}$ the rank revision, the hull revision and the extended hull revision respectively. Thus we dispose of three syntactic operators: $\diamond_{rk}, \diamond_h, \diamond_{eh}$ called rank arbitration operator, hull arbitration operator and extended hull arbitration operator respectively.

The following postulates are the natural translation of the postulates of Liberatore and Shaerf [24] for arbitration operators to our framework (SA states Syntactic Arbitration):

- (SA1) $P \diamond Q = Q \diamond P$
- (SA2) $P \diamond Q \subseteq P \odot Q$
- (SA3) If $P \odot Q \not\vdash \perp$ then $P \odot Q \subseteq P \diamond Q$
- (SA4) $P \diamond Q \vdash \perp$ if and only if $P \vdash \perp$ and $Q \vdash \perp$
- (SA5) If $C_{fc}(P_1) = C_{fc}(P_2)$ and $C_{fc}(Q_1) = C_{fc}(Q_2)$ then $P_1 \diamond Q_1 = P_2 \diamond Q_2$
- (SA6) $P \diamond (Q_1 \oplus Q_2) = \begin{cases} P \diamond Q_1 & \text{or} \\ P \diamond Q_2 & \text{or} \\ (P \diamond Q_1) \oplus (P \diamond Q_2) \end{cases}$
- (SA7) $P \oplus Q \subseteq P \diamond Q$
- (SA8) If $P \not\vdash \perp$ then $P \odot (P \diamond Q) \not\vdash \perp$

Postulate SA1 states that the two pieces of information have equal priority. Postulate SA2 says that “conjunction” is stronger than arbitration. Postulate SA3, together with SA2, say that under the consistency of the “conjunction” of programs, such a “conjunction” is the result of arbitration of programs. Postulate SA4 says that the only possibility for the inconsistency of arbitration is the inconsistency of each input. Postulate SA5 is the equivalence of the syntax in our context. Postulate SA6 is the trichotomy postulate in our context. Postulate SA7 says that arbitration is stronger than “disjunction”. Postulate SA8 says that the output of arbitration will be consistent with any consistent input.

The following result summarizes the behavior of our arbitration operators with respect to the postulates above defined.

Theorem 1 *The operators \diamond_{rk} , \diamond_h , \diamond_{eh} satisfy the postulates SA1, SA2, SA3, SA4, SA7 and SA8. They don't satisfy SA5 nor SA6.*

Proof:

(SA1) It is straightforward because of the definition of \diamond_{rk} , \diamond_h , \diamond_{eh} , since $P \diamond Q = C_{fc}(P \star Q) \cap C_{fc}(Q \star P)$ and the intersection is commutative.

(SA2) Since $C_{fc}(P \circ_{rk} Q) \subseteq C_{fc}(P \circ_h Q) \subseteq C_{fc}(P \circ_{eh} Q)$, it is enough to prove that \diamond_{eh} satisfies (SA2) to establish that \circ_{rk} and \circ_h satisfy also SA2. Let $I_P(Q) = \{H_1, H_2, \dots, H_n\}$ and $I_Q(P) = \{T_1, T_2, \dots, T_m\}$ where each H_i is a maximal subset of P containing $P_{\rho(P,Q)}$ and Q -consistent, and each T_j is a maximal subset of Q containing $Q_{\rho(Q,P)}$ and P -consistent. By definition, $P \diamond_{eh} Q = C_{fc}(P \circ_{eh} Q) \cap C_{fc}(Q \circ_{eh} P)$ where $C_{fc}(P \circ_{eh} Q) = \bigcap_{i=1}^n C_{fc}(H_i \cup Q)$ and $C_{fc}(Q \circ_{eh} P) = \bigcap_{j=1}^m C_{fc}(T_j \cup P)$. Since $H_i \subseteq P$, we have $H_i \cup Q \subseteq P \cup Q$; analogously $T_j \cup P \subseteq Q \cup P$. By the monotonicity of C_{fc} we have $P \diamond_{eh} Q \subseteq C_{fc}(P \cup Q) = P \odot Q$. Thus, SA2 is satisfied.

(SA3) Suppose $P \odot Q \not\vdash \perp$, that is $C_{fc}(P \cup Q) \not\vdash \perp$, so $P \cup Q$ is consistent. Then $\rho(P, Q) = 0$, that is $P_{\rho(P,Q)} = P$ and therefore $P \circ_{rk} Q = P \cup Q$. Analogously, $Q \circ_{rk} P = Q \cup P$; therefore $P \diamond_{rk} Q = C_{fc}(P \cup Q) = P \odot Q$. Thus, $P \odot Q \subseteq P \diamond_{rk} Q \subseteq P \diamond_h Q \subseteq P \diamond_{eh} Q$. That is, Postulate (SA3) is satisfied by our three operators.

(SA4) Suppose $P \diamond_{eh} Q \vdash \perp$. We want to see that $C_{fc}(P \circ_{eh} Q) \vdash \perp$ and $C_{fc}(Q \circ_{eh} P) \vdash \perp$. Consider $I_P(Q) = \{H_1, H_2, \dots, H_n\}$ and $I_Q(P) = \{T_1, T_2, \dots, T_m\}$ where each H_i is a maximal subset of P containing $P_{\rho(P,Q)}$ and Q -consistent, and each T_j is a maximal subset of Q containing $Q_{\rho(Q,P)}$ and P -consistent. We have $\bigcap_{i=1}^n C_{fc}(H_i \cup Q)$ is not consistent, therefore $C_{fc}(H_i \cup Q)$ is not consistent; in particular $C_{fc}(P_{\rho(P,Q)} \cup Q)$ is not consistent. By definition of \circ_{rk} , it follows that $P_{\rho(P,Q)} = \emptyset$. Thus $C_{fc}(Q) \vdash \perp$. With a similar argument, starting from $C_{fc}(Q \circ_{eh} P) \vdash \perp$ we obtain $C_{fc}(P) \vdash \perp$. This proves one direction of Postulate SA4, because if $P \diamond_{rk} Q \vdash \perp$ or $P \diamond_h Q \vdash \perp$ necessarily $P \diamond_{eh} Q \vdash \perp$.

Conversely, suppose that P and Q are inconsistent. We want to see that $P \diamond_{rk} Q \vdash \perp$, $P \diamond_h Q \vdash \perp$ and $P \diamond_{eh} Q \vdash \perp$. As before it is enough to see that $P \diamond_{rk} Q \vdash \perp$. By hypothesis, $C_{fc}(P) = Lit$ and $C_{fc}(Q) = Lit$, but $P \diamond_g Q =$

$C_{fc}(P_{\rho(P,Q)} \cup Q) \cap C_{fc}(Q_{\rho(Q,P)} \cup P)$. Moreover, $C_{fc}(P_{\rho(P,Q)} \cup Q) \supseteq C_{fc}(Q) = Lit$ and $C_{fc}(Q_{\rho(Q,P)} \cup P) \supseteq C_{fc}(P) = Lit$. Therefore $P \diamond_{rk} Q = Lit$.

(SA5) Our syntactic arbitration operators don't satisfy this postulate. We build a counterexample. Define $P_1 = \{a \rightarrow c; b\}$, $P_2 = \{b\}$, $Q_1 = \{b \rightarrow c; a\}$, $Q_2 = \{a\}$. It is clear that $C_{fc}(P_1) = C_{fc}(P_2) = \{b\}$ and $C_{fc}(Q_1) = C_{fc}(Q_2) = \{a\}$; since $P_i \cup Q_i$ is consistent for $i = 1, 2$, we have, by (SA2) and (SA3), $P_i \diamond Q_i = C_{fc}(P_i \cup Q_i)$ for $\diamond \in \{\diamond_{rk}, \diamond_h, \diamond_{eh}\}$. Notice that $P_1 \cup Q_1 = \{a \rightarrow c; b \rightarrow c; a; b\}$, thus $C_{fc}(P_1 \cup Q_1) = \{a, b, c\}$; also notice that $C_{fc}(P_2 \cup Q_2) = C_{fc}(\{b, a\}) = \{a, b\}$. Therefore our operators are syntax dependent.

(SA6) Our syntactic arbitration operators don't satisfy this postulate. We build a counterexample. Let's define $P = \{a \rightarrow b; a \rightarrow c; e\}$, $Q_1 = \{a\}$, $Q_2 = \{b\}$. Since $P \cup (Q_1 \oplus Q_2) = P$, $P \cup Q_1 = \{a \rightarrow b; a \rightarrow c; e; a\}$ and $P \cup Q_2 = \{a \rightarrow b; a \rightarrow c; e; b\}$ are consistent, we have for $\diamond \in \{\diamond_{rk}, \diamond_h, \diamond_{eh}\}$ the following equalities: $P \diamond (Q_1 \oplus Q_2) = C_{fc}(P) = \{e\}$, $P \diamond Q_1 = P \odot Q_1 = \{a, b, c, e\}$, $P \diamond Q_2 = P \odot Q_2 = \{b, e\}$ and $(P \diamond Q_1) \oplus (P \diamond Q_2) = \{b, e\}$. Thus it is clear that $P \diamond (Q_1 \oplus Q_2)$ is different from the options in the postulate.

(SA7) This postulate is verified by our three operators. Actually, $P \circ_{rk} Q = P_{\rho(P,Q)} \cup Q \supseteq Q$ and also $Q \circ_{rk} P \supseteq P$; from this it follows $C_{fc}(P) \subseteq C_{fc}(Q \circ_{rk} P)$ and $C_{fc}(Q) \subseteq C_{fc}(P \circ_{rk} Q)$ and by definition $P \oplus Q \subseteq P \diamond_{rk} Q \subseteq P \diamond_c Q \subseteq P \diamond_{eh} Q$.

(SA8) Assume that $C_{fc}(P) \neq Lit$. Consider $I_Q(P) = \{\{T_1, T_2, \dots, T_m\}$ where T_i is a maximal subset of Q containing $Q_{\rho(Q,P)}$ and P -consistent. Since P is consistent, $C_{fc}(T_i \cup P) \neq Lit$ for $i = 1, \dots, m$. Notice that $P \diamond_{eh} Q = (P \odot_{eh} Q) \oplus (Q \odot_{eh} P) \subseteq \bigcap_{i=1}^m C_{fc}(T_i \cup P)$. Then, by Lemma 1, $P \odot (P \diamond_{eh} Q) \subseteq C_{fc}(P \cup C_{fc}(T_i \cup P)) = C_{fc}(T_i \cup P)$. Therefore, $P \odot (P \diamond_{eh} Q) \neq Lit$ and necessarily $P \odot (P \diamond_{rk} Q) \neq Lit$ and $P \odot (P \diamond_c Q) \neq Lit$. ■

Now we illustrate the behavior of our three syntactic operators. In particular we shall see that they have behaviors well differentiated.

Example 3 Let P and Q be two programs defined as follows: $P = \{a, b \rightarrow \neg c; b \rightarrow d; b \rightarrow \neg c; \neg c \rightarrow e; a, \neg c \rightarrow f; a\}$ and $Q = \{a, b \rightarrow c; a \rightarrow e; a, e \rightarrow c; a, e \rightarrow d; c \rightarrow d; c \rightarrow f; b\}$. Then $C_{fc}(P) = \{a\}$ and $C_{fc}(Q) = \{b\}$. It is easy to verify that the bases of P and Q have two levels. No rule is exceptional, thus $P_1 = Q_1 = \emptyset$. Since $P \cup Q$ is not consistent, we have $P_{\rho(P,Q)} = Q_{\rho(Q,P)} = \emptyset$. Therefore $I_Q(P)$ is the set of maximal subsets of Q which are P -consistent. With a little computation, we can verify that $I_Q(P) = \{T_1, T_2\}$ where $T_1 = \{a \rightarrow e; a, e \rightarrow d; c \rightarrow d; c \rightarrow f; b\}$ and $T_2 = \{a, b \rightarrow c; a \rightarrow e; a, e \rightarrow c; a, e \rightarrow d; c \rightarrow d; c \rightarrow f\}$. Moreover $C_{fc}(T_1 \cup P) = \{a, b, \neg c, d, e, f\}$ and $C_{fc}(T_2 \cup P) = \{a, d, e\}$. For the same reasons as before, $I_P(Q)$ is the set of maximal subsets of P which are Q -consistent. This can be easily computed: $I_P(Q) = \{H_1, H_2\}$ where $H_1 = \{b \rightarrow d; \neg c \rightarrow e; a, \neg c \rightarrow f; a\}$ and $H_2 = \{a, b \rightarrow \neg c; b \rightarrow d; b \rightarrow \neg c; \neg c \rightarrow e; a, \neg c \rightarrow f\}$. Then, $C_{fc}(H_1 \cup Q) = \{a, b, c, d, e, f\}$ and

$C_{fc}(H_2 \cup Q) = \{b, \neg c, d, e\}$. Thus, $\cap I_Q(P) = \{a \rightarrow e; c \rightarrow d; c \rightarrow f\}$ and $\cap I_P(Q) = \{b \rightarrow d; \neg c \rightarrow e; a, \neg c \rightarrow f\}$. Therefore, $C_{fc}(\cap I_Q(P) \cup P) = \{a, e, d\}$ and $C_{fc}(\cap I_P(Q) \cup Q) = \{b, d\}$. Finally we can compute the outputs for the three operators. $P \diamond_h Q = (P \circ_h Q) \oplus (Q \circ_h P) = C_{fc}(P \circ_h Q) \cap C_{fc}(Q \circ_h P) = \{d\}$. Since $P_{\rho(P,Q)} \cup Q = Q$ and $Q_{\rho(Q,P)} \cup P = P$, we have $P \diamond_{rk} Q = C_{fc}(P) \cap C_{fc}(Q) = \emptyset$. For the last operator we have $P \diamond_{eh} Q = (P \circ_{eh} Q) \oplus (Q \circ_{eh} P) = C_{fc}(P \circ_{eh} Q) \cap C_{fc}(Q \circ_{eh} P)$ where $C_{fc}(P \circ_{eh} Q) = C_{fc}(H_1 \cup Q) \cap C_{fc}(H_2 \cup Q) = \{b, d, e\}$ and $C_{fc}(Q \circ_{eh} P) = C_{fc}(T_1 \cup P) \cap C_{fc}(T_2 \cup P) = \{a, d, e\}$. Therefore $P \diamond_{eh} Q = \{d, e\}$. Summarizing, we have:

$$P \diamond_{rk} Q = \emptyset \subset \{d\} = P \diamond_h Q \subset \{d, e\} = P \diamond_{eh} Q$$

4 Merging programs

Merging operators with integrity constraints were defined in [18]. Therein we can find a characterization in terms of postulates. The aim of this section will be to define merging operators for the representation of beliefs as programs as those presented in Section 2 with the logic of forward chaining defined therein. We shall also define the merging postulates in this syntactical framework and we shall study the relationships between our new operators and the postulates.

We denote by *Prog* the set of all the programs; $\mathcal{M}(\text{Prog})$ will denote the set of finite and nonempty multisets of nonempty programs. The merging operators Δ we are interested in, are operators from $\mathcal{M}(\text{Prog}) \times \text{Prog}$ into subsets of *Lit*. The multisets of programs are called profiles and we shall use the letters Φ and Ψ (with subscripts if necessary) to denote them. If $\Phi = \{P_1, \dots, P_n\}$ is a profile and P is a program, $\Delta(\Phi, P)$ must be understood as the result of merging the programs in Φ under the constraint P . We shall write $\Delta_P(\Phi)$ or $\Delta_P(P_1, \dots, P_n)$ instead of $\Delta(\Phi, P)$. For a profile Φ we denote $\cup \Phi$ the union of all programs in Φ . If Φ_1 and Φ_2 are profiles we denote by $\Phi_1 \sqcup \Phi_2$ the new profile resulting of the union of multisets (e.g. $\{P\} \sqcup \{P\} = \{P, P\}$).

Guided by Equation 2, and the interpretation of “disjunction” already defined, we set the following definition:

$$\Delta_P(P_1, \dots, P_n) = \begin{cases} C_{fc}(P \cup P_1 \dots \cup P_n), & \text{if this is consistent;} \\ \bigcap C_{fc}(P_i \circ_{rk} P), & \text{otherwise.} \end{cases} \quad (4)$$

where the profile is $\{P_1, \dots, P_n\}$ (the programs to merge), under the integrity constraint P and \circ_{rk} is the rank revision. We adopt the following definition of entailment for programs P and Q : we put $P \vdash Q$ whenever $C_{fc}(Q) \subset C_{fc}(P)$.

The following postulates are the adaptation of postulates characterizing merging operators with integrity constraints (see [18]):

Let $\Phi = \{P_1, \dots, P_n\}$.

(FP0) $\Delta_P(\Phi) \vdash P$

(FP1) If P is consistent, then $\Delta_P(\Phi)$ is consistent.

(FP2) If $C_{fc}(\cup \Phi \cup P)$ is consistent, then $\Delta_P(\Phi) = C_{fc}(\cup \Phi \cup P)$.

(FP3) If $\Phi_1 = \{P_1, \dots, P_n\}$, $\Phi_2 = \{Q_1, \dots, Q_n\}$, $C_{fc}(P_i) = C_{fc}(Q_i)$, $C_{fc}(P) = C_{fc}(Q)$, then $\Delta_P(\Phi_1) = \Delta_Q(\Phi_2)$

(FP4) If $P_1 \vdash P$, $P_2 \vdash P$ and P_1 and P_2 are consistent, then

$$C_{fc}(\Delta_P(P_1, P_2) \cup P_1) \neq Lit \Rightarrow C_{fc}(\Delta_P(P_1, P_2) \cup P_2) \neq Lit$$

(FP5) $\Delta_P(\Phi_1) \cup \Delta_P(\Phi_2) \vdash \Delta_P(\Phi_1 \sqcup \Phi_2)$

(FP6) If $\Delta_P(\Phi_1) \cup \Delta_P(\Phi_2)$ is consistent, then $\Delta_P(\Phi_1 \sqcup \Phi_2) \vdash \Delta_P(\Phi_1) \cup \Delta_P(\Phi_2)$

(FP7) $\Delta_P(\Phi) \cup Q \vdash \Delta_{P \cup Q}(\Phi)$

(FP8) If $\Delta_P(\Phi) \cup Q$ is consistent, then $\Delta_{P \cup Q}(\Phi) \vdash \Delta_P(\Phi) \cup Q$

Postulate FP0 means that the integrity constraint is respected. Postulate FP1 establishes the consistency of the merging whenever the integrity constraint are consistent. Postulate FP2 establishes that in the case where there is no conflict between the pieces of information, the output is the consequence of putting all pieces together. Postulate FP3 is the independence of the syntax in our framework. Postulate FP4 is the postulate of fairness. Postulates FP5 and FP6 refer to a good behavior of the merging of subgroups: if the merging of subgroups agree on some facts and it is consistent, the result of the merging on the whole group is the set of facts on which the subgroups agree. Postulates FP7 and FP8 concerning the iteration of the process (see [18] for more detailed explanations about the postulates).

The following theorem tells us how is the behavior of the operator defined by Equation (4) with respect the postulates in the case of ranked revision.

Theorem 2 *The operator Δ^{rk} satisfies the postulates FP0, FP1, FP2 and FP4. It doesn't satisfy FP3 nor FP5-FP8*

Proof: (FP0) This postulate follows straightforward from the definition: if $C_{fc}(P \cup P_1 \dots \cup P_n)$ is consistent, we have $\Delta_P(P_1, \dots, P_n) = C_{fc}(P \cup P_1 \dots \cup P_n) \supset C_{fc}(P)$; therefore $\Delta_P(P_1, \dots, P_n) \vdash P$. If $\Delta_P(P_1, \dots, P_n) = \bigcap C_{fc}(P_i \circ_{rk} P)$, by definition of rank revision (see Definition 4), $P_i \circ_{rk} P = P_i^{\rho(P_i, P)} \cup P \supset P$. Thus, $\Delta_P(P_1, \dots, P_n) = \bigcap C_{fc}(P_i \circ_{rk} P) \vdash P$.

(FP1) This postulate is straightforwardly verified, because $C_{fc}(P \cup P_1 \dots \cup P_n)$ is consistent or by the definition of rank revision $P_i \circ_{rk} P$ is consistent, so $\bigcap C_{fc}(P_i \circ_{rk} P)$ is consistent.

(FP2) This postulate is straightforward by the definition.

(FP3) This postulate is not verified. We show a counterexample. Consider $\Phi_1 = \{P_1\}$ and $\Phi_2 = \{Q_1\}$ where $P_1 = \{a \rightarrow b\}$ and $Q_1 = \{a \rightarrow c\}$. It is clear that $C_{fc}(P_1) = \emptyset = C_{fc}(Q_1)$. Define $P = Q = \{a\}$. Then $\Delta_P(P_1) = C_{fc}(P \cup P_1) = \{a, b\}$ and $\Delta_Q(Q_1) = \{a, c\}$. Thus, $\Delta_P(P_1) \neq \Delta_Q(Q_1)$.

(FP4) We need some technical results in order to prove that this postulate is verified.

Lemma 2 *Let Q, P be programs such that Q is consistent, $Q \cup P$ is inconsistent and $Q \vdash P$, then $P \vdash Q \circ_{rk} P$.*

Proof: Suppose there is $l \in C_{fc}(Q \circ_{rk} P)$ and $l \notin C_{fc}(P)$. Since $P \cup Q$ is inconsistent, we have $Q \circ_{rk} P = Q_i \cup P$ for $i \geq 1$. From the consistency of Q follows that Q_i has no facts. Let L_0, \dots, L_r be the hierarchy of $C_{fc}(Q_i \cup P)$ (see discussion before Example 1). In L_0 there are only literals of P . Let l be minimal in the hierarchy of $C_{fc}(Q_i \cup P)$ such that $l \notin C_{fc}(P)$. We claim that there exists a rule $l_1, \dots, l_n \rightarrow l$ in Q_i such that $\{l_1, \dots, l_n\} \subset C_{fc}(P)$. To establish the claim we proceed as follows: since l is minimal such that $l \notin C_{fc}(P)$, necessarily $l \notin L_0$, because $L_0 \subset C_{fc}(P)$. Let L_i be such that $l \in L_i$ and $l \notin L_k$ for $k < i$. By the minimality of l , it is clear that $L_0 \cup L_1 \cup \dots \cup L_{i-1} \subset C_{fc}(P)$. Moreover, there exists a rule $l_1, \dots, l_n \rightarrow l$ in $Q_i \cup P$ such that $\{l_1, \dots, l_n\} \subset L_0 \cup L_1 \cup \dots \cup L_{i-1}$; the rule $l_1, \dots, l_n \rightarrow l$ is not in P , otherwise $l \in C_{fc}(P)$; therefore $l_1, \dots, l_n \rightarrow l$ is a rule of Q_i . Moreover the rule $l_1, \dots, l_n \rightarrow l$ is exceptional in Q . But $C_{fc}(P) \subset C_{fc}(Q)$, thus, Q is inconsistent, because $l_1, \dots, l_n, l \in C_{fc}(Q)$. Contradiction. ■

Corollary 1 *If $C_{fc}(Q \circ_{rk} P) \cup Q$ is inconsistent, then $Q \cup P$ is inconsistent.*

Proof: If $Q \cup P$ is consistent, we have $Q \circ_{rk} P = Q \cup P$. Thus, $C_{fc}(C_{fc}(Q \circ_{rk} P) \cup Q) = C_{fc}(Q \cup P) \cup Q = C_{fc}(P \cup Q)$ the last equality because of Lema 1. ■

Corollary 2 *If $Q \vdash P$ and Q is consistent, then $C_{fc}(Q \circ_{rk} P) \cup Q$ is consistent.*

Proof: If $C_{fc}(Q \circ_{rk} P) \cup Q$ is inconsistent, by Corollary 1, we have $Q \cup P$ is inconsistent. Thus, all conditions of Lemma 2 are verified. Therefore, $C_{fc}(Q \circ_{rk} P) \subset C_{fc}(P)$. Since $C_{fc}(P) \subset C_{fc}(Q)$, we have $C_{fc}(Q \circ_{rk} P) \subset C_{fc}(Q)$. Thus, because Q is consistent, $C_{fc}(Q \circ_{rk} P) \cup Q$ is consistent, a contradiction. ■

Now we are ready to show that the operator defined by Equation (4) satisfies FP4. Suppose that $P \cup P_1 \cup P_2$ is consistent. In this case $\Delta_P(P_1, P_2) = C_{fc}(P \cup P_1 \cup P_2)$. Thus,

$$C_{fc}(\Delta_P(P_1, P_2) \cup P_2) = C_{fc}(C_{fc}(P \cup P_1 \cup P_2) \cup P_2) = C_{fc}(P \cup P_1 \cup P_2)$$

is consistent (last equality is given by Lemma 1). In this case we have

$$C_{fc}(\Delta_P(P_1, P_2) \cup P_2) = C_{fc}(\Delta_P(P_1, P_2) \cup P_1)$$

Suppose now $P \cup P_1 \cup P_2$ is inconsistent. By definition of the operator

$$\Delta_P(P_1, P_2) = C_{fc}(P_1 \circ_{rk} P) \cap C_{fc}(P_2 \circ_{rk} P)$$

If $\Delta_P(P_1, P_2) \cup P_2$ is inconsistent, it follows $C_{fc}(P_2 \circ_{rk} P) \cup P_2$ is inconsistent, because

$$\Delta_P(P_1, P_2) \cup P_2 = (C_{fc}(P_1 \circ_{rk} P) \cap C_{fc}(P_2 \circ_{rk} P)) \cup P_2 \subset C_{fc}(P_2 \circ_{rk} P) \cup P_2$$

and, by Corollary 1, we have $P_2 \cup P$ is inconsistent. Since P_2 is consistent and $P_2 \vdash P$, the hypotheses of Lemma 2 hold. Therefore, $P \vdash C_{fc}(P_2 \circ_{rk} P)$. But $P_2 \vdash P$, thus $P_2 \vdash C_{fc}(P_2 \circ_{rk} P)$. Therefore $C_{fc}(P_2 \circ_{rk} P) \cup P_2$ is consistent, a contradiction. Thus, $C_{fc}(P_2 \circ_{rk} P) \cup P_2$ is consistent, that is Postulate FP4 holds.

Actually, we have proved that if $P_1 \vdash P$, $P_2 \vdash P$ and P_1, P_2 are consistent, then $C_{fc}(P_2 \circ_{rk} P) \cup P_2$ and $C_{fc}(P_2 \circ_{rk} P) \cup P_1$ are consistent.

Postulates FP5-FP8 do not hold. We give counterexamples for each postulate.

Counterexample for FP5: $\Delta_P(\Phi_1) \cup \Delta_P(\Phi_2) \vdash \Delta_P(\Phi_1 \sqcup \Phi_2)$.

Consider $P = \{a\}$, $P_1 = \{a \rightarrow b\}$, $P_2 = \{b \rightarrow c\}$. Then $\Delta_P(P_1, P_2) = \{a, b, c\}$, $\Delta_P(P_1) = \{a, b\}$ y $\Delta_P(P_2) = \{a\}$. Thus, $\Delta_P(P_1) \cup \Delta_P(P_2) \not\vdash \Delta_P(P_1, P_2)$ because $\Delta_P(P_1, P_2) \not\subseteq \Delta_P(P_1) \cup \Delta_P(P_2)$.

Counterexample for FP6: If $\Delta_P(\Phi_1) \cup \Delta_P(\Phi_2)$ is consistent, then $\Delta_P(\Phi_1 \sqcup \Phi_2) \vdash \Delta_P(\Phi_1) \cup \Delta_P(\Phi_2)$.

Consider $P = \{a\}$, $P_1 = \{\neg c; a \rightarrow b\}$, $P_2 = \{b \rightarrow c\}$. Thus, $\Delta_P(P_1) = \{a, b, \neg c\}$ and $\Delta_P(P_2) = \{a\}$. Therefore $\Delta_P(P_1) \cup \Delta_P(P_2)$ is consistent. Since $C_{fc}(P_1 \cup P_2 \cup P)$ is inconsistent, we have $\Delta_P(P_1, P_2) = C_{fc}(P_1 \circ_{rk} P) \cap C_{fc}(P_2 \circ_{rk} P) = \{a\}$ and $\Delta_P(\Phi_1) \cup \Delta_P(\Phi_2) \not\subseteq \Delta_P(P_1, P_2)$.

Counterexample for FP7: $\Delta_P(\Phi) \cup Q \vdash \Delta_{P \cup Q}(\Phi)$.

Consider $P = \{c\}$, $Q = \{a\}$, $\Phi = \{a \rightarrow b\}$. Thus, $\Delta_P(\Phi) \cup Q = \{a, c\}$ and $\Delta_{P \cup Q}(\Phi) = \{a, b, c\}$. Therefore $\Delta_P(\Phi) \cup Q \not\vdash \Delta_{P \cup Q}(\Phi)$.

Counterexample for FP8: If $\Delta_P(\Phi) \cup Q$ is consistent, then $\Delta_{P \cup Q}(\Phi) \vdash \Delta_P(\Phi) \cup Q$.

Consider $\Phi = \{H\}$ where $H = \{a \rightarrow c; b \rightarrow \neg c\}$, $P = \{a\}$, $Q = \{b\}$. We have $\Delta_P(\Phi) = \{a, c\}$, so $\Delta_P(\Phi) \cup Q = \{a, b, c\}$ is consistent. Moreover $\Delta_{P \cup Q}(\Phi) = C_{fc}(\Phi \circ_{rk} (P \cup Q))$. Since P has no exceptional rules, we have $\Phi_{\rho}(\Phi, P \cup Q) = \emptyset$. Therefore $\Delta_{P \cup Q}(\Phi) = C_{fc}(\Phi \circ_{rk} (P \cup Q)) = C_{fc}(\Phi_{\rho}(\Phi, P \cup Q) \cup (P \cup Q)) = C_{fc}(P \cup Q) = \{a, b\}$. Thus, $\Delta_{P \cup Q}(\Phi) \not\vdash \Delta_P(\Phi) \cup Q$. ■

The rest of this section is devoted to the analysis of properties of merging operators defined in the style of Equation (4) but using hull revision operators and extended hull revision operators instead of rank revision. The operators are given in the following way:

$$\Delta_P^h(P_1, \dots, P_n) = \begin{cases} C_{fc}(P \cup P_1 \dots \cup P_n), & \text{if consistent;} \\ \bigcap C_{fc}(P_i \circ_h P), & \text{otherwise.} \end{cases} \quad (5)$$

$$\Delta_P^{eh}(P_1, \dots, P_n) = \begin{cases} C_{fc}(P \cup P_1 \dots \cup P_n), & \text{if consistent;} \\ \bigcap C_{fc}(P_i \circ_{eh} P), & \text{otherwise.} \end{cases} \quad (6)$$

The following theorem tells us about the behavior of the merging operators defined by Equations (5) and (6).

Theorem 3 *The operators Δ^h and Δ^{eh} satisfy the postulates FP0, FP1 and FP4. They don't satisfy FP3-FP8*

Proof: Since $Q \circ_{eh} P \vdash Q \circ_h P \vdash Q \circ_{rk} P$, the verification of Postulate FP0 by these operators is straightforward. Postulates FP1 and FP2 follow from definitions of hull revision and extended hull revision.

The given counterexamples for FP3, FP5, FP6, FP7 and FP8 in the case of merging using rank revision remain valid for our merging defined by Equations (5) and (6), because in the counterexamples rank revision, hull revision and extended hull revision coincide. Notice that in the counterexample for FP8, we have $I_\Phi(P \cup Q) = \{\{a \rightarrow c\}, \{b \rightarrow \neg c\}\}$. Thus $h_\Phi(P \cup Q) = \emptyset$, and therefore

$$\Delta_{P \cup Q}^h(\Phi) = C_{fc}(h_\Phi(P \cup Q) \cup (P \cup Q)) = C_{fc}(P \cup Q) = \{a, b\}$$

We have also $\Delta_P^h(\Phi) \cup Q = \Delta_P(\Phi) \cup Q = \{a, b, c\}$. Thus, $\Delta_{P \cup Q}^h(\Phi) \not\models \Delta_P^h(\Phi) \cup Q$. We have $\Phi \circ_{eh} (P \cup Q) = \langle \{a : b : a \rightarrow c\}, \{a : b : b \rightarrow \neg c\} \rangle$ thus

$$C_{fc}(\Phi \circ_{eh} (P \cup Q)) = C_{fc}(\{a : b : a \rightarrow c\}) \cap C_{fc}(\{a : b : b \rightarrow \neg c\}) = \{a, b, c\} \cap \{a, b, \neg c\} = \{a, b\}$$

Therefore $\Delta_{P \cup Q}^{eh}(\Phi) = C_{fc}(\Phi \circ_{eh} (P \cup Q)) = \{a, b\}$. Thus, in this example, we have $\Delta_{P \cup Q}(\Phi) = \Delta_{P \cup Q}^h(\Phi) = \Delta_{P \cup Q}^{eh}(\Phi) = \{a, b\}$ and $\Delta_P(\Phi) \cup Q = \Delta_P^h(\Phi) \cup Q = \Delta_P^{eh}(\Phi) \cup Q = \{a, b, c\}$. Therefore $\Delta_{P \cup Q}^{eh}(\Phi) \not\models \Delta_P^{eh}(\Phi) \cup Q$.

Postulate (FP4) is also problematic for the new operators. We give a counterexample for this postulate. The counterexample works in the two cases (hull revision and extended hull revision) because the two operators coincide.

Consider $P_1 = \{a; a \rightarrow d; a, d \rightarrow c\}$, $P_2 = \{a; b; c \rightarrow \neg b; a \rightarrow d\}$ and $P = \{a; b \rightarrow c; d \rightarrow c\}$. We prove that they verify the hypothesis of Postulate (FP4). Moreover $\Delta_P^*(P_1, P_2) \cup P_1$ is consistent, whereas $\Delta_P^*(P_1, P_2) \cup P_2$ is inconsistent, where $\star \in \{h, eh\}$.

$C_{fc}(P_1) = \{a, d, c\}$, $C_{fc}(P_2) = \{a, b, d\}$ and $C_{fc}(P) = \{a\}$; thus, $P_1 \vdash P$, $P_2 \vdash P$ and P_1 and P_2 are consistent. We compute $\Delta_P^h(P_1, P_2)$. Since $P_1 \cup P_2 \cup P$ is inconsistent (more precisely $P_2 \cup P$ is inconsistent) we have, by definition

$$\Delta_P^h(P_1, P_2) = C_{fc}(P_1 \circ_h P) \cap C_{fc}(P_2 \circ_h P)$$

But $P_1 \cup P$ is consistent, thus

$$C_{fc}(P_1 \circ_h P) = C_{fc}(P_1 \cup P) = \{a, d, c\}$$

Since $P_2 \cup P$ is inconsistent, we look for the exceptional rules of P_2 . We see that $c \rightarrow \neg b$ is the unique exceptional rule of P_2 . This rule is P -consistent. Now we compute $I_{P_2}(P)$, the maximal subsets of P_2 containing $c \rightarrow \neg b$ which are P -consistent; we observe that if $b, c \rightarrow \neg b \in M$, with $M \subset P_2$, necessarily $M \cup P$ is inconsistent (because by forward chaining we obtain $\neg b$). Thus we conclude that in the sets of $I_{P_2}(P)$ the element b does not appear. Notice that

$\{a; c \rightarrow \neg b; a \rightarrow d\}$ is the subset of P_2 such that it doesn't contain b and moreover is P -consistent. Thus we conclude

$$I_{P_2}(P) = \langle \{a; c \rightarrow \neg b; a \rightarrow d\} \rangle$$

That is, there exists a unique maximal subset of P_2 containing $b \rightarrow \neg c$ and P -consistent.

Since $h_{P_2}(P) = \cap I_{P_2}(P) = \{a; c \rightarrow \neg b; a \rightarrow d\}$ and $P_2 \circ_h P = h_{P_2}(P) \cup P$ we have $C_{fc}(P_2 \circ_h P) = C_{fc}(h_{P_2}(P) \cup P) = \{a, d, c\}$. Therefore $\Delta_P^h(P_1, P_2) = C_{fc}(P_1 \circ_h P) \cap C_{fc}(P_2 \circ_h P) = \{a, d, c\}$. Finally we have $C_{fc}(\Delta_P^h(P_1, P_2) \cup P_1) = \{a, d, c\}$ and $C_{fc}(\Delta_P^h(P_1, P_2) \cup P_2) = Lit$ because $b \in P_2$ and, since $c \rightarrow \neg b \in P_2$ and $c \in \Delta_P^h(P_1, P_2)$ we obtain by forward chaining $\neg b$.

Since $\Delta_P^{eh}(P_1, P_2) = C_{fc}(P_1 \circ_{eh} P) \cap C_{fc}(P_2 \circ_{eh} P)$ and $C_{fc}(P_1 \circ_h P) = C_{fc}(P_1 \circ_{eh} P) \cap C_{fc}(P_2 \circ_h P) = C_{fc}(P_2 \circ_{eh} P)$ we have $\Delta_P^h(P_1, P_2) = \Delta_P^{eh}(P_1, P_2)$. This shows that postulate (FP4) doesn't hold for merging operators defined with extended hull revision. ■

5 Concluding remarks

We have defined three syntactic arbitration operators. They satisfy Postulates (SA1)–(SA4) and (SA7)–(SA8) but they fail to satisfy Postulates (SA5)–(SA6). These operators seem to be a good compromise between rationality and computation, good properties and tractability. The operator \diamond_{rk} is polynomial because \circ_{rk} is clearly polynomial. The other operators have a higher computational complexity inherent to computation of maximal consistent sets.

Concerning merging with integrity constraints, we have also three syntactic operators. The operator defined using rank revision satisfies Postulates (FP0), (FP1), (FP2) and (FP4), whereas the merging operators defined using hull revision and extended hull revision satisfy only Postulates (FP0), (FP1) and (FP2). Thus in the case of merging with integrity constraints the operator Δ defined with rank revision is better than the two other operators both from the point of view of rational behavior as from the point of view of computational complexity.

In order to satisfy more rational properties, it will be interesting to explore other alternatives to definition given by Equation (4), in particular, the use of some kinds of aggregation functions (like majority) to produce the resulting facts (see for instance [20]) could lead to operators having a better behavior. It remains also to explore the exact relationships between properties of arbitration and merging operators defined in Equations (3), (4), (5) and (6) and the properties of revision operators used in those definitions.

References

1. C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

2. C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.
3. C. Baral, S. Kraus, J. Minker, and V. S. Subrahmanian. Combining knowledge bases consisting of first-order theories. *Computational Intelligence*, 8(1):45–71, 1992.
4. H. Bezzazi, S. Janot, S. Konieczny, and R. Pino Pérez. *Analysing rational properties of change operators based on forward chaining*, pages 317–339. Springer LNCS 1472, 1998.
5. N. Creignou, O. Papini, R. Pichler, and S. Woltran. Belief revision within fragments of propositional logic. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *KR*. AAAI Press, 2012.
6. J. P. Delgrande. Horn clause belief change: Contraction functions. In G. Brewka and J. Lang, editors, *KR*, pages 156–165. AAAI Press, 2008.
7. J. P. Delgrande and P. Peppas. Revising horn theories. In T. Walsh, editor, *IJCAI*, pages 839–844. IJCAI/AAAI, 2011.
8. J. P. Delgrande, T. Schaub, H. Tompits, and S. Woltran. Merging logic programs under answer set semantics. In P. M. Hill and D. S. Warren, editors, *ICLP*, volume 5649 of *Lecture Notes in Computer Science*, pages 160–174. Springer, 2009.
9. J. P. Delgrande and R. Wassermann. Horn clause contraction functions: Belief set and belief base approaches. In F. Lin, U. Sattler, and M. Truszczynski, editors, *KR*. AAAI Press, 2010.
10. T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision , updates, and counterfactuals. *Artificial Intelligence*, 57(2-3):227, 270 1992.
11. P. Gärdenfors. *Knowledge in flux*. MIT Press, 1988.
12. P. Gärdenfors, editor. *Belief Revision*. Cambridge University Press, 1992.
13. S. O. Hansson. What’s new isn’t always best. *Theoria*, pages 1–13, 1998. Special issue on non-prioritized belief revision.
14. J. Hué, O. Papini, and E. Würbel. Merging belief bases represented by logic programs. In C. Sossai and G. Chemello, editors, *ECSQARU*, volume 5590 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 2009.
15. J. Hué, E. Würbel, and O. Papini. Removed sets fusion: Performing off the shelf. In M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. M. Avouris, editors, *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 94–98. IOS Press, 2008.
16. H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR’91)*, pages 387–394, 1991.
17. H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.
18. S. Konieczny and R. Pino Pérez. Merging information under constraints: a logical framework. *Journal of Logic and Computation*, 12(5):773–808, 2002.
19. S. Konieczny and R. Pino Pérez. Logic based merging. *Journal of Philosophical Logic*, 40:239–270, 2011.
20. Sébastien Konieczny. On the difference between merging knowledge bases and combining them. In *Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR’00)*, pages 135–144, 2000.
21. J. Lang and Pierre Marquis. Resolving inconsistencies by variable forgetting. In *Proceedings of the eighth International Conference on Principles of Knowledge*

- Representation and Reasoning (KR'02)*, pages 239–250, Toulouse, France, 2002. Morgan Kaufmann Publishers, Inc.
22. M. Langlois, R. H. Sloan, B. Szörényi, and G. Turán. Horn complements: Towards horn-to-horn belief revision. In D. Fox and C. P. Gomes, editors, *AAAI*, pages 466–471. AAAI Press, 2008.
 23. P. Liberatore and M. Schaerf. A commutative operator for belief revision. In *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence*, pages 217–228, 1995.
 24. P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.
 25. J. Lin. Integration of weighted knowledge bases. *Artificial Intelligence*, 83(2):363–378, 1996.
 26. J. Lin and A. O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information System*, 7(1):55–76, 1998.
 27. J. Lin and A. O. Mendelzon. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*. Kluwer, 1999.
 28. B. Nebel. Belief revision and default reasoning: syntax-based approaches. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 417–428, 1991.
 29. J. Pearl. System z: a natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *Proceedings of the Third Conference on Theoretical Aspects of Reasoning About Knowledge (TARK'90)*, 1990.
 30. P. Z. Revesz. On the semantics of theory change: arbitration between old and new information. In *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Databases*, pages 71–92, 1993.
 31. P. Z. Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7(2):133–160, 1997.